

https://dspace.org/

Empirical comparison of vector bootstrap methods for multivariate scenario generation

Murray, Malcolm James

2012-07-19
http://hdl.handle.net/10539/11711

Downloaded from a DSpace repository. https://dspace.org/

EMPIRICAL COMPARISON OF VECTOR BOOTSTRAP METHODS FOR MULTIVARIATE SCENARIO GENERATION

Malcolm James Murray

A research report submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg, 2011

ABSTRACT

Stochastic simulation models require input scenarios, which may be generated from observed data using bootstrap methods. If a model's input variables are auto- and/or cross-correlated, these dependencies must be preserved in the generated scenarios. Three bootstrap methods were tested empirically: (1) the vector moving block bootstrap method, with a block length of one timeframe, (2) the vector moving block bootstrap method, with an optimized block length, and (3) the vector nearest neighbour bootstrap method. They were applied to data observed from processes at a petro-chemical plant: 28 numerical, multivariate, stationary time series, with a variety of auto- and cross-correlations. The quality of the generated scenarios was measured using a Turing test procedure, which balances fidelity to the observed data and natural variety. Method (2) performed best, followed by method (3), and then method (1). The number of input variables bootstrapped simultaneously did not significantly affect the performance of the bootstrap methods.

DECLARATION

I hereby declare the following:

- This research report is my own, unaided work except where I have explicitly indicated otherwise (following the required referencing conventions).
- It is being submitted in partial fulfilment of the requirements for the Degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg.
- It has not been submitted before for any degree or examination at any other university.
- It contains data obtained while working under the aegis of Sasol Technology, Secunda, South Africa.

Signed

_____

Malcolm James Murray

this_____ day of _____ year _____

TABLE OF CONTENTS

ABSTRACT i
DECLARATION ii
TABLE OF CONTENTS iii
LIST OF FIGURES iv
LIST OF TABLES v
NOMENCLATURE vi
1 INTRODUCTION 1
1.1 Background: Input Scenarios for Simulation Models 1
1.2 Motivation and Scope 2
2 LITERATURE SURVEY 4
2.1 Types of Methods 4
2.2 Non-Parametric Bootstrap Methods 5
2.3 The Importance of Preserving Dependencies 11
2.4 The Vector Moving Block Bootstrap Method (VMB) 11
2.5 The Vector Nearest Neighbour Bootstrap Method (VNN) 12
3 OBJECTIVES 13
4 EXPERIMENTAL DESIGN AND METHOD 14
4.1 Dependent Variable – A Quality Measure 14
4.2 Influencing Factors 17
4.3 Hypotheses 17
4.4 Sampling 18
4.5 Computations 18
5 OBSERVATIONS 19
6 RESULTS 22
6.1 The Effect of the Number of Variables in the Trace Series 22
6.2 The Effect of the Method Used 25
7 DISCUSSION 30
8 CONCLUSIONS 34
9 RECOMMENDATIONS FOR FURTHER RESEARCH 35
REFERENCES 36
APPENDIX A: DIGITAL APPENDIX 38
APPENDIX B: R CODE FOR BOOTSTRAP METHODS 39
APPENDIX C: QUADE MULTIPLE COMPARISON TESTS 42

LIST OF FIGURES

Figure 1 - An example trace series with no auto- or cross-correlation 5
Figure 2 - An example trace series with cross-correlation but no auto-correlation 6
Figure 3 - An example trace series with auto-correlation but no cross-correlation 6
Figure 4 - The block bootstrap re-sampling approach 7
Figure 5 - An example trace series with auto- and cross-correlation 10
Figure 6 - Sample distributions of Q at various factor levels 23
Figure 7 - Distributions of ANOVA residuals for the three bootstrap methods 26
Figure 8 - ANOVA residuals vs predictions 27
Figure 9 - Differences in Q between bootstrap methods 28
Figure 10 - Example univariate trace series 31
Figure 11 - Generated series by the VMB_1 bootstrap method (block length of 1 timeframe) 31
Figure 12 - Generated series by the VMB bootstrap method (block length of 25 timeframes) 31
Figure 13 - Performance versus block length for an example trace subseries 32

LIST OF TABLES

Table 1 - Panel data format 2
Table 2 - Q observations: 2 variables in trace series 19
Table 3 - Q observations: 4 variables in trace series 20
Table 4 - Q observations: 8 variables in trace series 21
Table 5 - Sample statistics - Q 24
Table 6 - Overall Kruskal-Wallis results 25
Table 7 - Sample statistics - residuals 27
Table 8 - Multiple comparison tests results 29
Table 9 - Within-subjects data format 42

NOMENCLATURE

VMB	the vector moving block bootstrap method
VMB_1	the vector moving block bootstrap method, with a block length of 1 timeframe
VMB_{opt}	the vector moving block bootstrap method, with an optimized block length
VNN	the vector nearest neighbour bootstrap method
$\mathrm{HOC(k)}$	the k^{th} higher order crossing count of a univariate time series
b	the block length (the number of timeframes sampled by block bootstrap methods)
c	the number of levels of an experimental factor
D	the test statistic for the Kolmogorov-Smirnov test
F	the test statistic for the Quade test
g	the number of experimental subjects, or blocks
K	the number of candidates considered by nearest neighbour bootstrap methods
m	the number of timeframes in each subseries
n	the number of timeframes in the trace series
p	the calculated probability of rejecting a true null hypothesis
P_j	the probability of selecting the j^{th} candidate in the VNN method
q	the required number of timeframes in the generated series
Q	the quality measure for quantifying the performance of a bootstrap method
r_c	the Spearman rank cross-correlation coefficient
r_a	the Spearman rank auto-correlation coefficient (lag-1)
s	the number of subseries for the Turing test
α	the significance level of a statistical test (the acceptable type I error rate)
Δ	the delta measure of the difference in auto-correlation structures between two univariate time series
μ	the population median
χ^2	the chi-squared statistic from the Kruskal-Wallis test

1 INTRODUCTION

1.1 Background: Input Scenarios for Simulation Models

Stochastic simulation modelling is used to study the behaviour of complex systems, as follows:

1. A model is built, which is a simplified representation of the system that contains its essential features. The complexity of the model depends on the scope and the required accuracy, which depends on the purpose of the modelling exercise.
2. Scenarios are generated, which are sets of inputs to the model of the system. Again, this depends on the purpose of the modelling. If we want to do 'stress-testing' on the system, we can generate extreme scenarios from imagination. If we only need to estimate average system response, we can just use average values for all the inputs (having measured them for a while). However, we usually want to study the full range of response of the system under realistic conditions. This requires large volumes of realistic scenarios that mimic the natural level and variation of the inputs (which have been observed). We focus on this intermediate case.
3. The scenarios are fed into the model, which simulates the behaviour of the system.
4. The response of the system model is observed and analyzed statistically.

To get meaningful results, the model must be appropriate and the input scenarios must be plentiful and realistic. Ideally, all the required input scenarios would be observed empirically, sampling directly from the population of all possible input scenarios in the real world. However, it is seldom practical to observe them in sufficient quantities. Typically, only a modest empirical sample is available, and this must be used to generate the required number of input scenarios. This process of generating realistic input scenarios from observed data is the focus of this research.

The measured data is called trace data, and can be visualised as a panel (or matrix), where each column represents one input variable, and each row represents the timeframe during which the input variables were measured, as illustrated in Table 1. The scenarios generated from the measured data also have this panel format. These two panels contain multivariate time-series data, so they are called the *trace series* and the *generated series* respectively.

Table 1 - Panel data format

	Input variable 1	Input variable 2	Input variable 3	Input variable 4	etc.
Timeframe 1					
Timeframe 2					
Timeframe 3					
Timeframe 4					
Timeframe 5					
etc.					

1.2 Motivation and Scope

The complexity of each input variable in the trace series may vary as follows:

1. Data type:
    - Numerical (interval, ratio, discrete numerical, ordinal)
    - Nominal (categorical, binary)
    - A combination of these in one trace series
2. Time-variation:
    - Stationary – The statistical distribution is constant over the length of the series. In other words, if the series were divided into a few sections, they would have similar histograms.
    - Non-stationary – The statistical distribution changes over the length of the series. It may follow a trend or vary cyclically.
3. Self-dependency:
    - No auto-correlation – Successive values are independent.
    - Short memory auto-correlation – Each value depends on the previous few values.
    - Long-memory auto-correlation – Each value depends on many previous values.
4. Inter-dependency with other variables:
    - No cross-correlation – Each variable is independent.
    - Simple cross-correlation – When one variable's value is high, the other variable's value is high (or when one variable's value is high, the other variable's value is low).
    - Complex dependency – One variable's value depends on another variable's previous values, or on a combination of other variables' values.

The current research is motivated by the need to find a suitable method for generating realistic input scenarios for simulations of complex processes at a petro-chemical plant. The input variables are physical quantities like temperatures, pressures, concentrations and yields, measured on a continuous numerical scale. Observed data for trace series are in no short supply, with measurements taken automatically at regular intervals. The processes are fairly stable, with input variables fluctuating around a steady mean. The input variables almost always display significant self-dependency and inter-dependency. It is very important that these dependencies are reproduced in the generated input scenarios. In fact, the validity of simulation results critically depends on it. Although long-memory auto-correlations and complex inter-dependencies may exist, a workable solution is sought that can handle short-memory auto-correlation and simple inter-dependencies. With this context in mind, we formally limit our scope to include only input variables with the following properties:

- Data type: numerical
- Time-variation: stationary
- Self-dependency: short-memory auto-correlation (or no auto-correlation)
- Inter-dependency: simple cross-correlation (or no cross-correlation)

2 LITERATURE SURVEY

2.1 Types of Methods

The methods for generating input scenarios from observed data fall into two categories:

1. *Parametric* methods use the observed data to try to understand and model the underlying data-generating process driving the inputs. A mathematical model is fitted to the observed data in the trace series, and then used to generate new input scenarios.
2. *Non-parametric* methods, by contrast, do not try to understand the observed data, but just to mimic it. Non-parametric *bootstrap* methods, specifically, generate input scenarios by drawing samples randomly from the empirical sample. This *re-sampling* is done with replacement, so that the observed scenarios can be sampled repeatedly as input scenarios. In essence, non-parametric bootstrap methods try to replay the trace series in a realistic way.

The features, benefits and pitfalls of each category of methods are discussed at length by Barton *et al* (2002) and summarised below.

Parametric methods have the advantage that they can generate new scenarios that have not been observed before, so they are preferable for very short trace series. Non-parametric bootstrap methods cannot produce as much variety because they re-sample from observed data directly. They need longer trace series, so that there is a wide range of scenarios from which to sample.

A major disadvantage of parametric methods is that one has to find an appropriate model for the underlying process. This can be difficult if the underlying process is complicated or poorly understood. A poor model may lead to implausible scenarios being generated. By contrast, non-parametric bootstrap methods can be applied without making assumptions about the underlying process. They always produce realistic scenarios because they re-sample data that has actually been observed.

The quest for adequate models leads to another disadvantage of parametric methods: complexity. Sophisticated models require many model parameters to be calculated, and this can quickly become computationally infeasible. In addition, complex models are risky in practice. They can easily be misapplied by practitioners who lack rigorous statistical training, because they do not fully understand the complicated statistical techniques involved and the associated assumptions. On the other hand, non-parametric bootstrap methods typically mimic the observed data in simple, intuitive ways.

The petro-chemical plant, which provides the motivation for this research, is a data-rich environment. Since this nullifies the main comparative advantage of parametric methods, non-parametric bootstrap

methods become the obvious choice. Thus, this research focuses only on them. They are referred to simply as bootstrap methods (although they exclude the parametric bootstrap method, which is model-based).

2.2 Non-Parametric Bootstrap Methods

Different bootstrap methods are appropriate for different trace series, depending on their correlation structures. The bootstrap methods available for a number of cases are surveyed below. Each case is illustrated with a figure, which shows an example trace series containing two variables (Figure 1, Figure 2, Figure 3 and Figure 5). The lag-1 auto-correlation is shown for each variable (r_a), as well as the cross-correlation between the variables (r_c). The Spearman rank correlation coefficient is used, which is the Pearson correlation coefficient applied to ranked data. A value of 1 means perfect positive correlation, 0 means no correlation, and -1 means perfect negative correlation.

2.2.1 Case 1: Each variable in the trace series is independent (no auto- or cross-correlations)

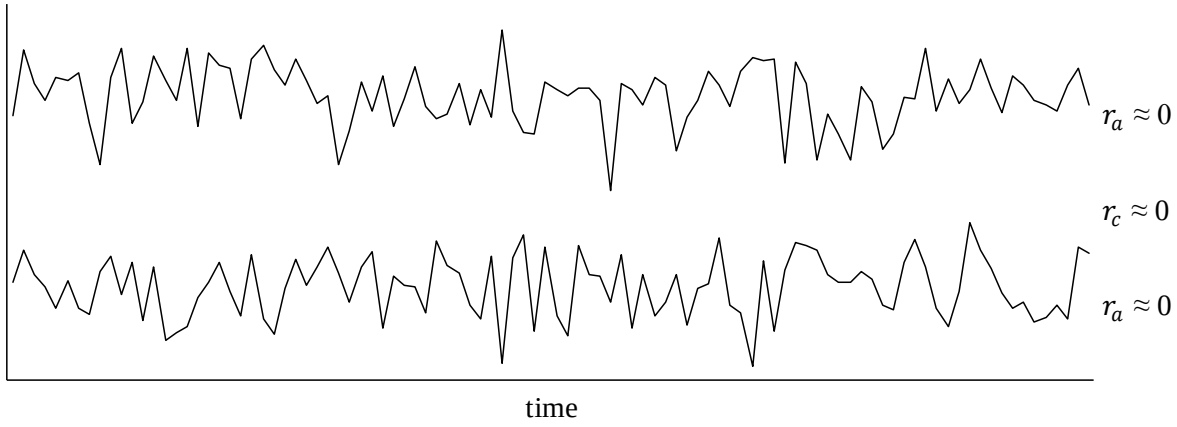


Figure 1 - An example trace series with no auto- or cross-correlation

In this case, the simple univariate bootstrap method introduced by Efron (1979) may be used. For each variable separately, a new series is generated by re-sampling randomly (with replacement) from the observed data in the trace series. Note: The standard parametric method for this case is to fit a parametric distribution to the observed data for each variable, and to generate new data by sampling randomly from the fitted distributions.

2.2.2 *Case 2: The variables in the trace series are cross-correlated, but not auto-correlated*

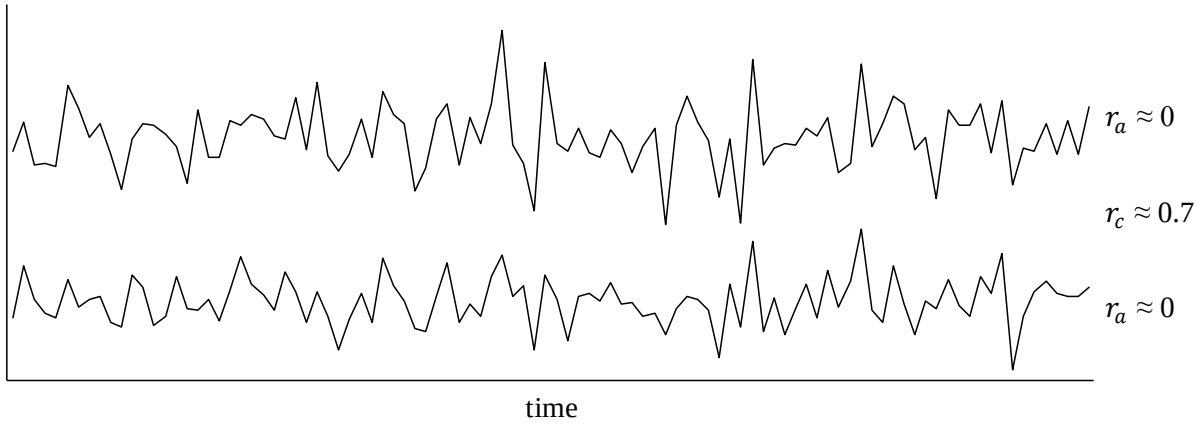


Figure 2 - An example trace series with cross-correlation but no auto-correlation

The simple univariate bootstrap method cannot be used in this case, because if the variables are re-sampled separately, their inter-dependencies will be destroyed in the generated series. A simple way to avoid this problem is to sample all the variables at once, as a vector. This automatically preserves the inter-dependencies, but at the expense of variety: only those combinations of values observed in the trace series can occur in the generated series, while many others may in fact be possible. Another approach is to perform a factor analysis on the variables and bootstrap them in the common factor space (Zientek & Thompson 2007). Note: An appropriate parametric method for this case is the NORTA method proposed by Cario & Nelson (1997).

2.2.3 *Case 3: The variables in the trace series are auto-correlated, but not cross-correlated*

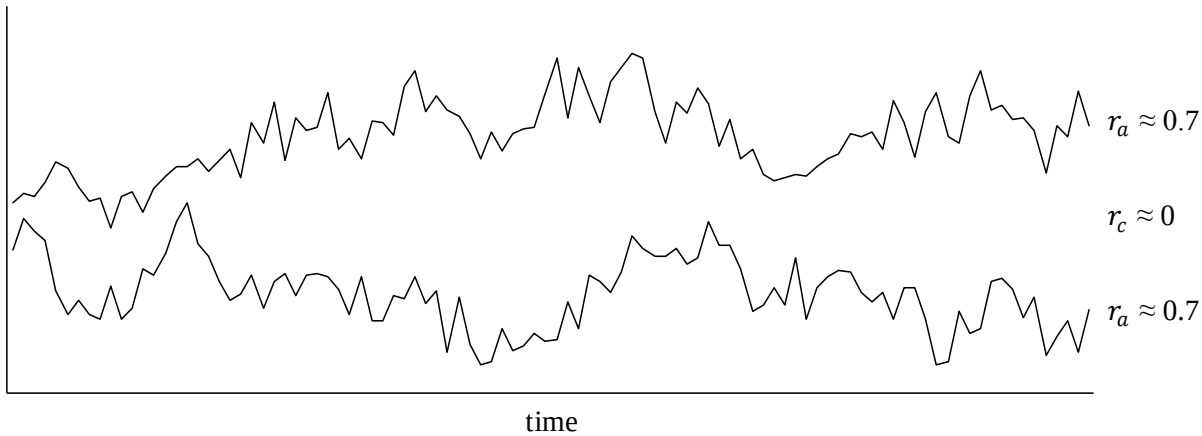


Figure 3 - An example trace series with auto-correlation but no cross-correlation

Again, the simple univariate bootstrap cannot be used in this case, because if samples are taken in random order, the auto-correlations will be destroyed in the generated series. There are two main approaches to overcoming this problem:

1. The *block* approach involves sampling a number of successive timeframes from the trace series at once, instead of just one at a time. The idea is that the auto-correlation pattern is preserved within each block, although there are discontinuities at the block boundaries in the generated series, as shown in Figure 4.

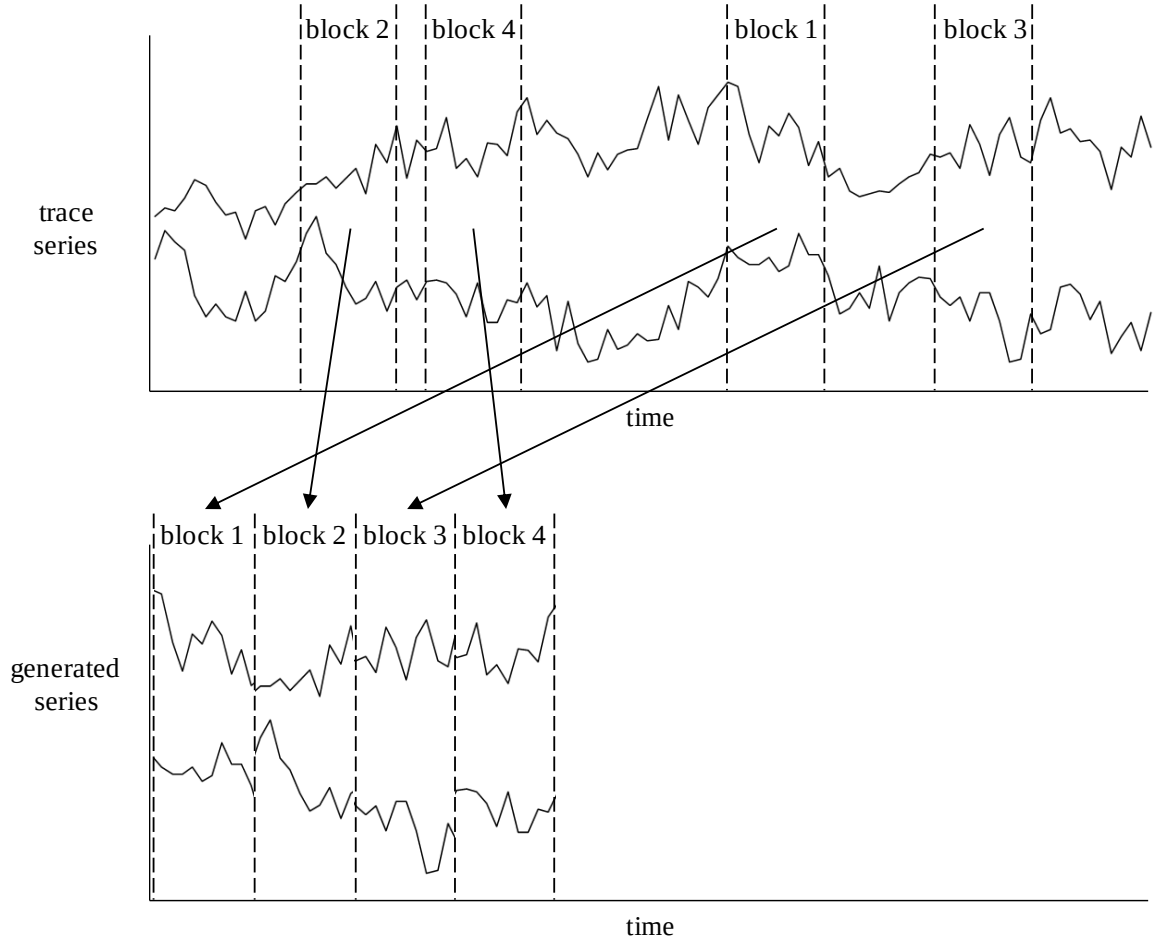


Figure 4 - The block bootstrap re-sampling approach

2. The *nearest neighbour* approach tries to preserve the auto-correlation by re-sampling such that each new sample 'fits' after the previous one. Each time it samples it asks, "What does the generated series look like right now?" and, "When did it look similar to this in the trace series", and then, "At those times, what happened next?" It then samples from one of these candidate successors.

The block bootstrap method comes in many varieties:

- The non-overlapping block bootstrap by Carlstein (1986) samples non-overlapping blocks of fixed length.
- The moving block bootstrap by Kunsch (1989) and Liu & Singh (1992) samples overlapping blocks of fixed length.

- The circular block bootstrap by Politis & Romano (1992) modifies the moving block bootstrap by looping the trace series around on itself so that the last timeframe leads on to the first.
- The stationary block bootstrap by Politis & Romano (1994) samples blocks whose length is sampled from a geometric distribution.
- The threshold block bootstrap by Park & Willemain (1999) samples non-overlapping blocks whose length is chosen such that each block begins and ends at a place where the trace series crosses its mean.
- The matched block bootstrap by Carlstein *et al* (1998) samples blocks so that they match at their ends using Markov chains.

A theoretical comparison of the first four methods by Lahiri (1999) shows that for a given block length, all the methods have the same amount of bias asymptotically, but differ in terms of variance, as follows:

- Methods that use overlapping blocks are better than those that use non-overlapping blocks.
- The circular block bootstrap and the moving block bootstrap are equivalent.
- The moving block bootstrap is better than the stationary block bootstrap.

Lahiri (1999) also compares his results with Carlstein *et al* (1998) and concludes that the matched block bootstrap performs best of all when block lengths are optimal. However, the matched block bootstrap is far more complex and computationally expensive. Willemain *et al* (2003) recommend the moving block bootstrap "because of its simplicity, prominence, and good performance". We follow their recommendation and choose the moving block bootstrap (with its circular modification) as the most suitable block bootstrap method for our purposes.

The method has one parameter, the block length, which is the number of timeframes sampled from the trace series at once. Hall *et* al (1995) show that the performance of the moving block bootstrap depends critically on block size. Demirel & Willemain (2002b) find that the optimal block size for scenario generation is proportional to the square root of the trace length, where the proportionality constant depends on the auto-correlation structure of the trace series. Lahiri (2004) provides a thorough survey of the available methods for estimating optimal block length, but these are designed for statistic-estimation rather than scenario generation.

The methods that follow the nearest neighbour approach (nearest neighbour methods) differ in the way they choose each new sample, as follows:

- *Number of previous timeframes compared* – When matching the 'current state' of the generated series with the trace series, a number of recent timeframes may be compared. The longer the memory of the auto-correlation structure, the more timeframes need to be compared. Lee & Ouarda (2011) propose a method for choosing this number based on the Akaike information criterion, but since our scope is limited to short memory auto-correlation, we choose to compare only the most recent timeframe.
- *Number of candidates considered* – All the timeframes in the trace series are ranked according to how well they resemble the current state of the generated series. Only the successors of the best matches are considered as candidates. The number of candidates considered may be chosen using:
    - an objective criterion proposed and tested by Lall & Sharma (1996) and supported by Rajagopalan & Lall (1999)
    - an approach proposed by Lee & Ouarda (2011) based on the Akaike information criterion
    - the following heuristic:

      $$number\ of\ candidates\ = \sqrt{number\ of\ timeframes\ in\ the\ trace\ series} \quad (1)$$

  Lall & Sharma (1996) and Rajagopalan & Lall (1999) have observed that the rule of thumb works well in practice when the trace series has more than 100 timeframes and no more than six previous timeframes are compared for matching. The trace series encountered at the petro-chemical plant are typically quite long, and we are only comparing one timeframe, so we choose the rule of thumb for its simplicity.
- *Successor selection method* – The successor (next sample) can be chosen from the candidates randomly, or by weighting the candidates so that the more closely the neighbour resembles the current state of the generated data, the better the chances of its successor being chosen. We choose the weighting scheme advocated by Lall & Sharma (1996) for its simplicity and effectiveness.
    - Let the number of candidates be *K*.
    - Rank these candidates from "best match" to "worst match".
    - Assign a probability P_j to the j^{th} candidate as follows: $P_j = \frac{1/j}{\sum_{i=1}^{K}(1/i)}$. (2)
    - Sample from these candidates with probabilities P_j.

Note that because nearest neighbour methods choose each new generated timeframe based on the previous ones, the first few generated timeframes must be sampled randomly from trace series. This warm-up period is discarded once the generated series is complete. Note also that an appropriate

parametric method for this case is the ARTA method proposed by Biller & Nelson (2005), with the ARTAFIT fitting algorithm (Biller & Nelson 2008).

2.2.4 Case 4: The variables in the trace series are both auto-correlated and cross-correlated

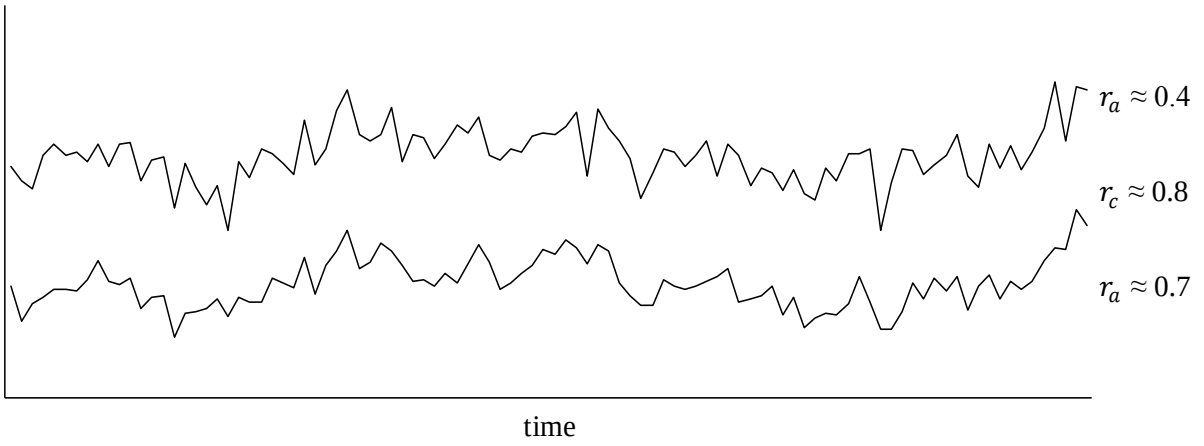


Figure 5 - An example trace series with auto- and cross-correlation

This is the combination of Case 2 and Case 3. It is a simple matter to generalize the moving block bootstrap method so that it is suitable for multivariate trace series. Instead of blocks being re-sampled for each variable separately, all the variables are re-sampled together, in a vector block. The nearest neighbour method can also be generalized for multivariate trace series with the vector-sampling approach; however, a complication is introduced: to find nearest neighbours, the 'current state' of the generated series must be compared to the trace series for all the variables at once. A vector distance measure is required. Teknomo (2006) provides a thorough survey of the many vector distance measures that are available for various data types. With possible future research in mind, we use a ranked weighted city block distance measure because it can be applied to trace series with mixed data types.

Suppose we have a vector representing the current state of the generated series, called the *feature vector*, and a number of *candidate vectors* from the trace series. We identify which candidate vectors are most similar to the feature vector as follows:

- Calculate the absolute difference between each variable in the feature vector and the corresponding variable in each of the candidate vectors.
- For each variable, rank the candidate vectors from smallest difference to largest difference.
- For each candidate vector, calculate the weighted sum of the ranks of all its variables, weighting according to strength of each variable's auto-correlation (calculated from the trace series), because it is most important to preserve the strongest auto-correlations.
- The lower this summed rank, the closer the candidate vector is to the feature vector.

Because this method compares distances for each variable before combining them for each vector, no standardization is required between variables, even if they are on different scales or of different types.

Note: An appropriate parametric method for this case is the copula-based multivariate time-series input model (which includes VARTA as a special case) proposed by Biller (2009).

2.3 The Importance of Preserving Dependencies

Sometimes it can be safely assumed (using statistical tests, judgement and experience) that the trace series is free from auto- and/or cross-correlations, or that they are unimportant for the simulation model at hand. At these times the simpler bootstrap methods presented for Cases 1, 2 and 3 should be used, because although the more sophisticated bootstrap methods will work for simpler cases, they require more computation time and sacrifice some variety. However, if there is any doubt, the more sophisticated bootstrap methods for Case 4 should be used. Neglecting relevant dependencies can have a serious impact on simulation results. In some cases, it can render the findings completely invalid. Civelek *et al* (2009) provide an exhaustive survey of empirical, analytic and simulation research showing that:

- Auto- and cross-correlations occur widely in practice in industries like manufacturing, services and telecommunications.
- Neglecting these dependencies when generating input scenarios can have a major effect on simulation outputs (such as utilization and waiting time for queuing systems).

In the following sections, algorithms are given for the two bootstrap methods mentioned for Case 4.

2.4 The Vector Moving Block Bootstrap Method (VMB)

Given a trace series with p variables and n timeframes, produce a generated series of any length as follows:

1. Choose a block length, b.
2. Randomly choose a timeframe from the n timeframes in the trace series.
3. Starting at this timeframe, sample a vector block from the trace series (p variables $\times$ b timeframes). If the sampling block overlaps the end of the trace series, let it continue at the beginning.
4. Repeat steps 2 and 3 until the generated series exceeds the required length.
5. Truncate the generated series to the exact length required.

2.5 The Vector Nearest Neighbour Bootstrap Method (VNN)

Given a trace series with p variables and n timeframes, produce a generated series of any length as follows:

1. For each variable, calculate the Spearman correlation between each value in the trace series and its previous value. This is the lag-1 auto-correlation for that variable (r_a).
2. Sample a timeframe randomly from the trace series. Set this as the start of the generated series. It serves as the 'warm-up' period.
3. Then, for each subsequent timeframe in the generated series:
    - Compare the previous timeframe in the generated series to every timeframe in the trace series, calculating the absolute difference between each of the variables.
    - For each variable, rank the timeframes in the trace series according to this absolute difference, in ascending order.
    - For each timeframe, calculate the weighted sum of these ranks, weighting each variable according the magnitude of its r_a.
    - Rank the timeframes in the trace series according to this weighted sum, in ascending order.
    - Choose randomly from the best $\sqrt{n}$ candidate timeframes (rounded down to the nearest whole number), according to the probabilities defined by Lall & Sharma (1996), as described in section 2.2.3.
    - Sample the successor timeframe of the chosen candidate as the new timeframe in the generated series.

3 OBJECTIVES

The objective of the research is to compare the VMB and VNN bootstrap methods to determine which method generates the most realistic scenarios for simulations at the petro-chemical plant. The testing is empirical, using trace series that have been observed from the plant.

The VMB bootstrap method has one parameter, the block length, which has a profound impact on performance. We consider two cases:

- Block length of one timeframe (VMB_1) - This is the simplistic case appropriate for when there is no auto-correlation in the trace series. We consider it as a control, because it is the method currently used at the petro-chemical plant.
- Optimized block length (VMB_{opt}) - The VMB bootstrap method is applied to each trace series at every possible block length. The block length with the best overall performance for a given trace series is chosen.

So, we want to compare the performance of three bootstrap methods: VMB_1, VMB_{opt} and VNN.

4 EXPERIMENTAL DESIGN AND METHOD

4.1 Dependent Variable – A Quality Measure

To compare the performance of the bootstrap methods objectively, we need a quantitative statistical measure of the quality of the generated scenarios. High quality scenarios are realistic scenarios. To be realistic, they must satisfy two criteria:

1. Fidelity - Obviously, the generated scenarios must be similar to the observed data, with the same essential characteristics. The properties of each variable in the trace series must be preserved in the generated series, namely its:
    - statistical distribution
    - auto-correlation structure
    - cross-correlation structure
2. Variety - However, the generated scenarios must not be identical to the observed data, because the process creating the trace data is stochastic not deterministic. Successive realizations of this process would produce different observed data. Therefore, the generated data must also have some realistic variation. (To illustrate the point, imagine that the whole trace series was just replayed repeatedly to make the generated series. The statistics would then be identical, but there would be no variety. This is certainly not realistic.)

An appropriate quality measure must reconcile these two competing objectives. Demirel & Willemain (2002b) note that many studies measuring bootstrap performance fail to do so, and propose that the correct approach is to phrase the quality measure as a Turing test (Demirel & Willemain 2002a). The idea behind a Turing test is as follows. Say you are shown Series A, which has been observed from some process. You are then shown Series B and C. Series B has been observed independently from the same process, while Series C has been generated from Series A using a bootstrap method. If you can't tell which is which, between Series B and C, then the generated series has a high quality, showing both fidelity and variety. In other words, a high quality generated series is as similar to the trace series as independent realizations of the trace series are to each other.

The VMB and VNN bootstrap methods do not attempt to achieve variety in the statistical distributions of the trace series, because they resample from the trace series directly. This is the disadvantage of all bootstrap methods compared with parametric methods. We assume that the two methods do achieve fidelity, because they only resample observed data, and do so in random ways.

Similarly, both methods sacrifice variety regarding cross-correlations, in favour of fidelity. They both sample each timeframe in the generated series at once, as a vector, from the trace series. This approach means that only the specific combinations of variables that have been observed in the trace series can occur in the generated series. This is not ideal, because we want to expose the simulation model to as wide a range of potential input scenarios as possible. However, the problem is not so serious if the trace series is long, because it already contains a variety of combinations. This is typically the case at the petro-chemical plant. Although not ideal, the vector approach seems sufficient for our purposes.

Regarding auto-correlations, the VMB and VNN bootstrap methods are both able to achieve a balance between fidelity and variety. The VMB achieves the trade-off by varying the block length. The longer the block length, the better the fidelity. The shorter the block length, the better the variety. An optimal block length can be found for each individual trace series. The VNN achieves the trade-off by the way it chooses from the candidate timeframes in the trace series. Variety is introduced by selecting randomly from the best $\sqrt{n}$ candidates. Fidelity is sought by weighting the selection probabilities so that the candidates that match best have the greatest chance of being sampled.

We want to compare the two methods according to how realistically they reproduce the auto-correlations *for each variable* in the trace series. We perform a Turing test by answering three questions:

1. How similar are independent realizations of the trace series to each other?
2. How similar are the generated series to the trace series?
3. How closely do these similarities match?

To answer questions 1 and 2 for each variable in the series, we must quantify the difference in auto-correlation structure between two univariate series. Auto-correlation functions and spectra may be used, but Demirel & Willemain (2002b) recommend the delta (Δ) difference measure because of its resistance to outliers. The measure is based on the *higher order crossings* (HOC) of each series, which are calculated as follows:

- Subtract the series mean from the series. Count the number of times this mean-centred series crosses the zero-line, to give the first higher order crossing count, HOC(1).
- Calculate the difference between successive timeframes in the mean-centred series. This gives the *first difference* series. Count the number of times it crosses the zero-line, to give HOC(2).
- Calculate the difference between successive timeframes in the first difference series. This gives the *second difference* series. Count the number of times it crosses the zero-line, to give HOC(3).

- Repeat up to HOC(10), which is sufficient to characterize short-memory time series for scenario-generation purposes (Demirel & Willemain 2002b).
- Note: If a difference series never crosses zero, set the HOC to 1. If it crosses zero at every timeframe, subtract 1 from the HOC. These adjustments are made to avoid dividing by zero in subsequent calculations.

The Δ difference measure between series i and series j (each n timeframes long) is then defined as:

$$\Delta = \sum_{k=1}^{10} \left(\frac{d_{ki} - d_{kj}}{SE\{d_{ki} - d_{kj}\}} \right)^2 \quad (3)$$

where: $d_{ki} = HOC(k)/(n-k)$ for series i

$d_{kj} = HOC(k)/(n-k)$ for series j

$$SE\{d_{ki} - d_{kj}\} = \sqrt{\frac{d_{ki}(1-d_{ki}) + d_{kj}(1-d_{kj})}{n-k}}$$

Simulation experiments by Demirel & Willemain (2002b) show that Δ follows a log-normal distribution.

Unfortunately, a problem arises in answering question 1 for real-world simulation scenarios, because we seldom have multiple independent realizations of the trace series. We usually just have one, and we want to get maximum value from it. Demirel & Willemain (2002b) propose getting around this problem by dividing the trace series into a number of subseries, and comparing these to each other. We use an adaptation of their method, answering the three Turing test questions as follows.

We have one observed trace time series, with a length of n timeframes, containing p variables. We then:

1. Divide the trace series into s non-overlapping subseries of equal length (m timeframes long), where $s = \sqrt{n}$ (rounded down to the nearest whole number) and m is the quotient of n and s. Compare each subseries to every other subseries, calculating Δ for each pair of subseries. This results in $\binom{s}{2}$ samples of Δ for each variable. These samples estimate the true distribution of Δ for each variable.
2. From each of the s subseries, generate s new series of length m using a bootstrap method. Compare each new generated series to its trace subseries, again calculating Δ for each variable. This results in s^2 samples of Δ for each variable. These samples estimate the generated distribution for each variable.
3. Compare the true and generated distributions for each variable, calculating the Kolmogorov-Smirnov D-statistic. For each variable, calculate the quality measure Q, which we define as $Q = ln[(1-D)/D]$ for good statistical properties. The higher the Q value, the better the

distributions match, the better the quality of the generated series for that variable. (Note: The natural range of D is [0,1]. To give Q a finite range, we limit the range of D to [0.018,0.982]. This gives Q a range of [-4,4].)

Thus Q serves as the measure by which performance is quantified, and is the dependent variable in the experimentation.

4.2 Influencing Factors

Our scope is limited to stationary trace series containing numerical data, with short-memory auto-correlation and simple cross-correlation. Other factors that may affect the quality of generated scenarios, for a given trace series are discussed below.

- The number of variables in the trace series. In the experimentation, we vary it between three levels: 2, 4 and 8 variables. Each trace series that is observed in the experimentation can take only one of these levels, so this is a between-subjects factor. (These levels were just chosen for convenience, to make good use of the available data.)
- The bootstrap method used – In the experimentation, we vary it between three levels: VMB_{opt}, VNN, and VMB_1. Each method can be applied to each observed trace series, so this is a within-subjects factor.
- Random effects arising from the individual nature of each trace series, and from the randomness in the method algorithms.

Note:

- The bootstrap methods deal with timeframes in a general way, so the quality does not depend on whether they represent seconds, minutes or hours, etc.
- The length of the trace series cannot affect quality, because the quality measures are defined in comparison to the trace series.

4.3 Hypotheses

We want to test whether performance depends on the number of variables in the trace series (for a given method), so we test the null hypothesis $H_{1,0}$*: Q does not depend on the number of variables in the trace series* against the alternative hypothesis $H_{1,A}$*: Q depends on the number of variables in the trace series* for each method separately.

More importantly, we want to test whether performance depends on the method used (for a given number of variables in the trace series), so we test the null hypothesis $H_{2,0}$*: Q does not depend on the bootstrap method that is used* against the alternative hypothesis $H_{2,A}$*: Q depends on the bootstrap*

method that is used. If $H_{1,0}$ is not rejected for any method, we can combine all the different numbers of variables to test $H_{2,0}$. Otherwise, $H_{2,0}$ must be tested separately for each number of variables. Note: All hypothesis tests are done at the 5% significance level.

4.4 Sampling

Empirical trace series were observed from processes at the petro-chemical plant, as follows:

- 16 separate trace series containing 2 variables each (32 variables in total)
- 8 separate trace series containing 4 variables each (32 variables in total)
- 4 separate trace series containing 8 variables each (32 variables in total)

Because each variable gives an estimate of Q, this yields 32 estimates for each level of the between-subjects factor (the number of variables in the trace series). Thus the design is balanced.

The trace series were selected by a chemical engineer working at the plant (Branca 2011) such that the variables in each trace series are:

- known to be linked in a physical way
- likely to be used as simultaneous inputs to a simulation model
- more-or-less stationary time series (from cursory visual inspection)

We hope that the total sample of 96 variables includes a fair variety of correlation structures, but we cannot guarantee that it is truly representative. Furthermore, the samples only come from one field – petro-chemical processes - so the applicability of the findings is limited to that field, strictly speaking. However, this restriction can be cautiously relaxed if inspection of data from other fields shows similar behaviour.

The variables in the sampled trace series represent physical measurements such as temperatures, pressures, flow rates, etc. The data are supplied digitally on a compact disc in Appendix A. To protect the intellectual property of the petro-chemical company, no names are given and each variable has been standardised by subtracting the mean and dividing by the standard deviation. This does not affect the correlations or result in any loss of generality.

4.5 Computations

The R language for statistical computing was used to perform the required computations. The script is included digitally in Appendix A and printed in Appendix B. Q was calculated for each variable in each trace series.

5 OBSERVATIONS

Table 2, Table 3 and Table 4 show the Q values that were calculated for each variable in each trace series, for each bootstrap method. The first two columns index the variable and trace series to which the bootstrap methods were applied. The next column gives the value of the between-subjects factor (the number of variables in the trace series). The final three columns give the Q values for each level of the within-subjects factor (the method used).

Table 2 - Q observations: 2 variables in trace series

Variable no	Trace series no	Number of variables in trace series	VMB_1	VMB_{opt}	VNN
1	1	2	-1.46	2.85	-0.12
2	1	2	0.89	1.33	-0.75
3	2	2	-3.20	0.96	0.63
4	2	2	-4.00	2.52	-0.95
5	3	2	-2.36	1.32	-0.13
6	3	2	-1.00	0.18	-0.08
7	4	2	-3.48	2.59	0.54
8	4	2	-2.78	1.28	1.32
9	5	2	-4.00	2.93	-0.90
10	5	2	-3.10	0.86	0.44
11	6	2	-0.29	2.15	0.93
12	6	2	1.83	1.56	0.90
13	7	2	-0.55	0.93	1.15
14	7	2	-4.00	-1.30	-1.92
15	8	2	1.64	0.13	1.41
16	8	2	-2.11	2.26	1.67
17	9	2	-0.07	1.32	2.18
18	9	2	0.15	1.77	1.56
19	10	2	-1.81	1.60	0.79
20	10	2	-2.17	1.54	0.73
21	11	2	-2.05	1.64	-0.07
22	11	2	-2.19	1.02	0.60
23	12	2	-0.79	0.68	0.78
24	12	2	-0.99	0.91	-0.92
25	13	2	-2.23	-0.11	-0.24
26	13	2	-1.92	1.45	0.97
27	14	2	-3.49	2.74	0.12
28	14	2	2.76	0.51	-0.40
29	15	2	0.55	0.19	0.89
30	15	2	-3.28	2.49	-0.93
31	16	2	-0.26	0.86	0.27
32	16	2	-2.51	3.02	-0.12

Table 3 - Q observations: 4 variables in trace series

Variable no	Trace series no	Number of variables in trace series	VMB_1	VMB_{opt}	VNN
33	17	4	-1.42	0.37	1.15
34	17	4	-4.00	-1.10	-1.11
35	17	4	-2.74	1.55	0.51
36	17	4	-2.18	2.64	-0.49
37	18	4	-1.91	0.77	0.89
38	18	4	-0.02	1.26	1.57
39	18	4	-0.52	1.84	1.48
40	18	4	-0.48	1.97	1.96
41	19	4	-2.15	1.42	0.52
42	19	4	-1.65	1.68	1.09
43	19	4	-2.83	1.31	0.74
44	19	4	-1.84	0.97	0.73
45	20	4	-1.93	1.55	0.83
46	20	4	2.21	1.43	-1.04
47	20	4	-3.09	0.88	0.20
48	20	4	-2.63	1.27	0.73
49	21	4	-1.00	0.81	1.07
50	21	4	-1.23	2.81	-0.05
51	21	4	-2.98	-1.26	-1.40
52	21	4	-3.42	1.63	1.15
53	22	4	-0.34	1.26	1.46
54	22	4	-0.36	0.79	0.61
55	22	4	-0.61	1.53	1.58
56	22	4	-4.00	0.16	0.42
57	23	4	2.27	1.72	-0.33
58	23	4	0.71	2.14	-2.05
59	23	4	-0.06	1.45	-1.53
60	23	4	1.35	1.87	-0.79
61	24	4	-1.32	1.41	0.77
62	24	4	-1.22	0.66	0.57
63	24	4	-2.21	0.51	0.27
64	24	4	-0.94	0.87	0.71

Table 4 - Q observations: 8 variables in trace series

Variable no	Trace series no	Number of variables in trace	VMB_1	VMB_{opt}	VNN
65	25	8	-0.74	0.90	0.56
66	25	8	-4.00	1.58	0.30
67	25	8	-1.16	1.73	2.00
68	25	8	-0.23	-0.09	0.67
69	25	8	0.43	1.02	0.67
70	25	8	-1.20	0.53	-0.22
71	25	8	-1.60	1.40	0.75
72	25	8	-1.69	0.57	-0.45
73	26	8	-0.19	1.58	1.29
74	26	8	0.73	3.04	0.65
75	26	8	0.16	1.81	1.67
76	26	8	0.05	1.91	1.60
77	26	8	-1.78	-0.04	0.67
78	26	8	-0.04	2.08	1.51
79	26	8	-1.06	0.57	2.05
80	26	8	0.27	1.55	1.91
81	27	8	-3.12	1.54	0.09
82	27	8	-2.63	2.13	0.54
83	27	8	-4.00	1.15	-0.16
84	27	8	2.66	1.04	-1.03
85	27	8	-4.00	1.11	-0.65
86	27	8	-0.65	-0.19	0.74
87	27	8	-1.03	0.72	1.10
88	27	8	-0.89	0.52	1.54
89	28	8	-0.04	2.30	0.61
90	28	8	1.03	2.12	-0.27
91	28	8	0.32	2.22	0.31
92	28	8	-0.14	1.22	-1.97
93	28	8	-0.31	1.86	0.34
94	28	8	-0.99	2.11	1.83
95	28	8	1.71	1.17	0.47
96	28	8	1.05	1.78	-0.45

6 RESULTS

6.1 The Effect of the Number of Variables in the Trace Series

We want to determine if and how the dependent variable (Q) is affected by the between-subjects factor (the number of variables in the trace series). This question must be answered for each method separately. The appropriate parametric test is one-way analysis of variance (ANOVA), which makes a number of assumptions about the data. In general, the assumptions involve residuals, which are the differences between the observed values and those predicted by the model, but for one-way ANOVA, the assumptions can be tested from the data directly, as follows:

- The dependent variable must be normally distributed at each level of the between-subjects factor.
- The standard deviation of the dependent variable must be similar at all the levels of the between-subjects factor, with a maximum difference factor of 2.
- There must be no serious outliers or influential points.

Figure 6 shows box-and-whisker plots of the observed Q values separately for each method, for each number of variables. In each plot, the bold line shows the median, which gives an indication of the central tendency of the data. The "box" shows the inter-quartile range, which gives and indication of the spread (or variance) of the data. The "whiskers" extend beyond the box edges no more than 1.5 times the box length, and the outliers beyond the whiskers are shown as circles.

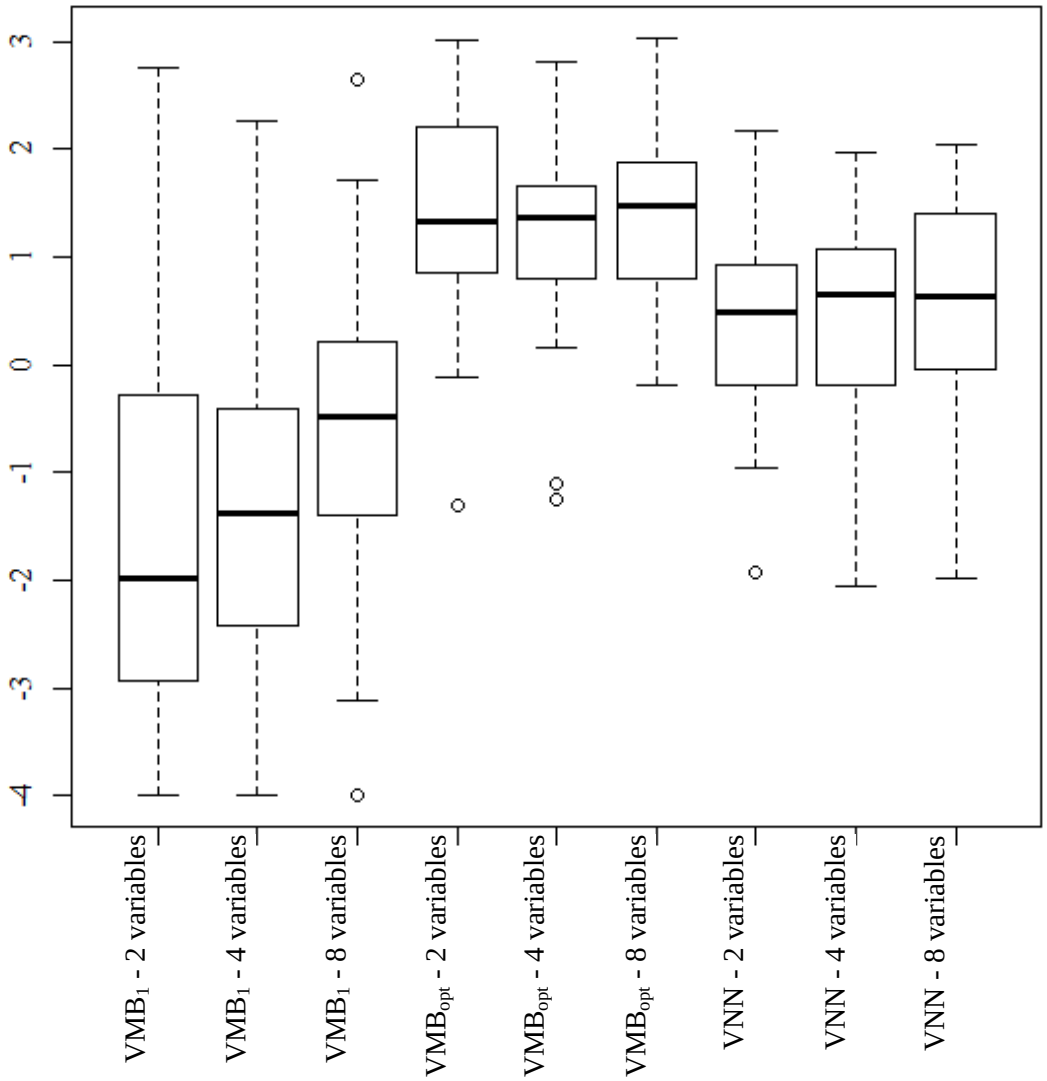


Figure 6 - Sample distributions of Q at various factor levels

From these plots we can see that:

- For each method, the spread of the data is fairly similar between the numbers of variables.
- Some of the groups' distributions are quite skew (e.g. VMB_1 - 2 variables, VMB_{opt} - 2 variables, and VNN - 4 variables). However, there does not seem to be a consistent direction or pattern to the skewness.
- For each method, there are small differences in median performance between the number of variables. When viewed in the context of the spread of the data, these differences seem negligible.

Table 5 shows the sample mean, standard deviation, median and inter-quartile range for each group. A Kolmogorov-Smirnov test was done for each group to test the null hypothesis that the data comes from a normal distribution, using the *ksnormTest()* function from the *fBasics* package in R. The two-sided p-values from these tests are also given in the table.

Table 5 - Sample statistics - Q

Group	Method	Number of variables	Sample mean	Sample standard deviation	Sample median	Sample inter-quartile range	Kolmogorov-Smirnov p-value (2-sided)
1	VMB_1	2	-1.5	1.8	-2.0	2.6	3.7×10^{-08}
2	VMB_1	4	-1.3	1.6	-1.4	1.9	7.0×10^{-07}
3	VMB_1	8	-0.7	1.6	-0.5	1.5	3.7×10^{-02}
4	VMB_{opt}	2	1.4	1.0	1.3	1.3	5.3×10^{-11}
5	VMB_{opt}	4	1.2	0.9	1.4	0.8	3.6×10^{-10}
6	VMB_{opt}	8	1.3	0.8	1.5	1.0	1.0×10^{-11}
7	VNN	2	0.3	0.9	0.5	1.1	1.1×10^{-01}
8	VNN	4	0.4	1.0	0.7	1.2	2.2×10^{-03}
9	VNN	8	0.6	0.9	0.6	1.3	1.0×10^{-03}

From the table we see that for each method, the standard deviation is similar between the numbers of variables, with a maximum difference factor of 1.25 (between groups 4 and 6). However, 8 of the 9 groups show non-normality when tested at the 5% significance level, and some show serious non-normality (very small p-values). Since there is no obvious pattern to the skewness, it is unlikely that normality can be achieved by means of a transformation. Therefore, we cannot use one-way ANOVA, and must resort to a non-parametric test to examine the effect of the number of variables (for each method separately). The Kruskal-Wallis rank sum test provides an appropriate alternative because:

- the number of variables is a between-subjects factor with more than two levels, and
- the spread of Q is similar at these three levels (for each method separately).

We rephrase the null hypothesis more specifically as $H_{1,0}$: *The median Q does not depend on the number of variables in the trace series* ($\mu_{2variables} = \mu_{4variables} = \mu_{8variables}$). The alternative hypothesis is then $H_{1,A}$*: The median Q depends on the number of variables in the trace series* ($\mu_{2variables} \neq \mu_{4variables}$ *or* $\mu_{2variables} \neq \mu_{8variables}$ *or* $\mu_{4variables} \neq \mu_{8variables}$).

These hypotheses were tested for each method separately using the *overall* Kruskal-Wallis test, which was applied in R using the *kruskal.test()* function from the *stats* package. The results are shown in Table 6.

Table 6 - Overall Kruskal-Wallis results

Method	Kruskal-Wallis chi-squared (χ^2)	p-value (2-sided)
VMB_1	5.87	0.053
VMB_{opt}	0.56	0.755
VNN	0.79	0.673

The p-values for the VMB_{opt} and VNN bootstrap methods are much greater than 0.05, so we fail to reject $H_{1,0}$ for them. We find no evidence to suggest that *the median Q depends on the number of variables in the trace series* for these methods. The p-value for the VMB_1 method, however, is only just greater than 0.05. We provisionally fail to reject $H_{1,0}$ for this method also, but note that this failure might be because the Kruskal-Wallis test does not have enough power at these sample sizes. (It is difficult to evaluate the power of the test because it is non-parametric, and we do not know the exact underlying distribution of the data.)

6.2 The Effect of the Method Used

Because we find no significant effect on Q by the number of variables, we can combine all the observed trace series when testing the effect of the bootstrap method used. We thus have 96 observed trace series, to which all three bootstrap methods have been applied. We wish to test $H_{2,0}$ to determine whether any of the methods is better than the others. The bootstrap method used is a within-subjects factor, so the appropriate parametric test is repeated-measures ANOVA. This equates to a two-way ANOVA with each subject treated as a separate level of an index factor. In this case, we must test the assumptions on the residuals, as follows:

- The residuals must be normally distributed at each level of the within-subjects factor.
- The standard deviation of the residuals must be similar at all the levels of the within -subjects factor, with a maximum difference factor of 2.
- There must be no serious outliers or influential points.

In addition, repeated-measures ANOVA has the assumption of *sphericity*, which requires that the differences in Q between all possible pairs of levels of the within-subjects factor have similar standard deviations.

A repeated-measures ANOVA model (type III) was fitted to the observed data using the R function *ezANOVA()* from the *ez* package, and the residuals were calculated. Figure 7 shows box-and-whisker plots of the residuals for each method. From the plots we can see that:

- The spread of the data is fairly similar between the methods.
- The distributions seem to be symmetrical for VMB_1 and VMB_{opt}, but negatively skewed for VNN.
- All three methods show some outliers.

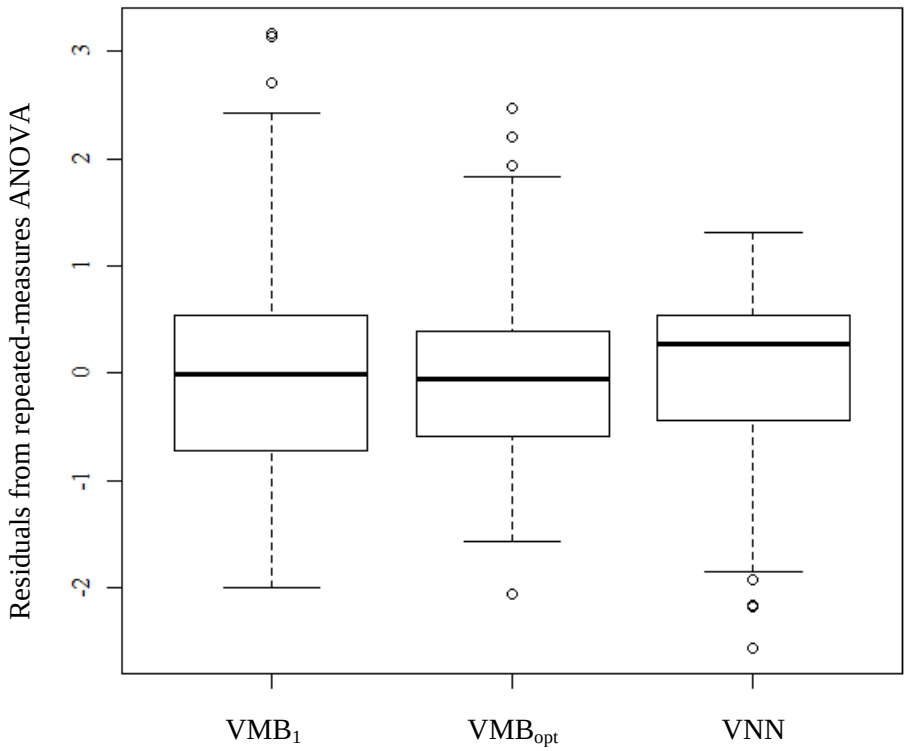


Figure 7 - Distributions of ANOVA residuals for the three bootstrap methods

Table 7 shows the standard deviation of the residuals for each method. Again, a Kolmogorov-Smirnov test was done for each method's residuals to test the null hypothesis that the data comes from a normal distribution. As before, the tests were done in R using the *ksnormTest()* function from the *fBasics* package. The two-sided *p*-values are given in the table.

Table 7 - Sample statistics - residuals

Method	Standard deviation	Kolmogorov-Smirnov *p*-value (2-sided)
VMB_1	1.15	0.653
VMB_{opt}	0.84	0.126
VNN	0.84	0.004

From the table we see that the standard deviation is similar between the methods, with a maximum difference factor of 1.4. However, the distribution for the VNN method shows non-normality when tested at the 5% significance level, with a *p*-value of 0.004. It is unlikely that normality can be achieved by means of a transformation, because the residuals follow normal distributions already for the other methods. This is confirmed by a plot of residuals versus predicted values in Figure 8 which shows no obvious pattern to suggest an appropriate transformation.

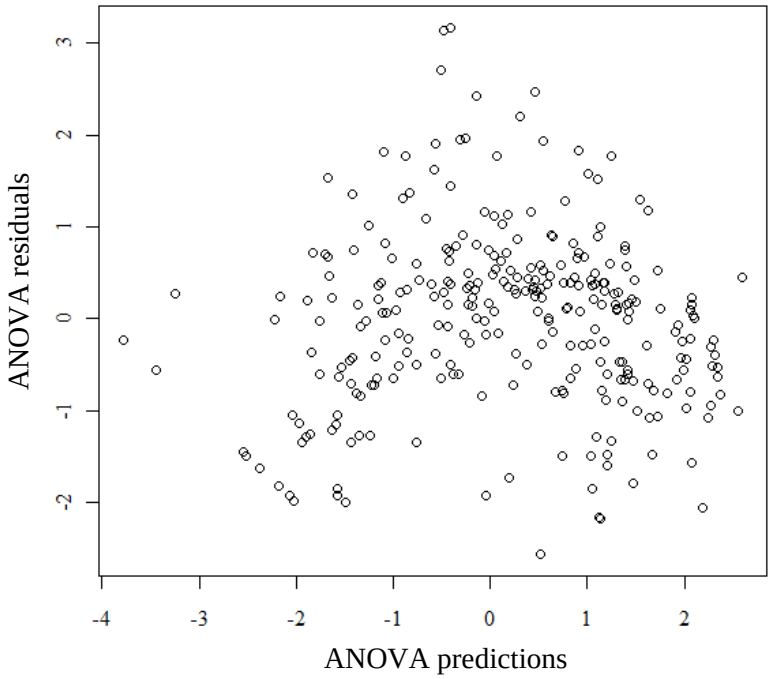


Figure 8 - ANOVA residuals vs predictions

Because the normality assumption is violated, we cannot use repeated-measures ANOVA. No sphericity tests are necessary. We must resort to a non-parametric test instead, to compare the bootstrap methods. Two alternatives are available for testing the effect of a within-subjects factor with more than two levels: the Friedman test and the Quade test. Conover (1999) states that the more powerful test is Quade if the within-subjects factor has 4 or fewer levels, and Friedman otherwise. Since we have only three methods, we use the overall Quade test. We rephrase the null hypothesis more specifically as $H_{2,0}$: *There is no median difference in* Q *between the bootstrap methods* ($\mu_{(VMBopt\ -VMB1)} = 0\ and\ \mu_{(VNN-VMB1)} = 0\ and\ \mu_{(VMBopt\ -VNN)} = 0$). The alternative hypothesis is then $H_{2,A}$*: There is a median difference in* Q *between the bootstrap methods* ($\mu_{(VMBopt\ -VMB1)} \neq 0\ or\ \mu_{(VNN-VMB1)} \neq 0\ or\ \mu_{(VMBopt\ -VNN)} \neq 0$).

Figure 9 shows box-and-whisker plots of the differences in Q between the three bootstrap methods. We see a pronounced difference between VMB_{opt} and VMB_1, and between VNN and VMB_1, but it is not clear whether there is a significant difference between VMB_{opt} and VNN. We expect to reject $H_{2,0}$.

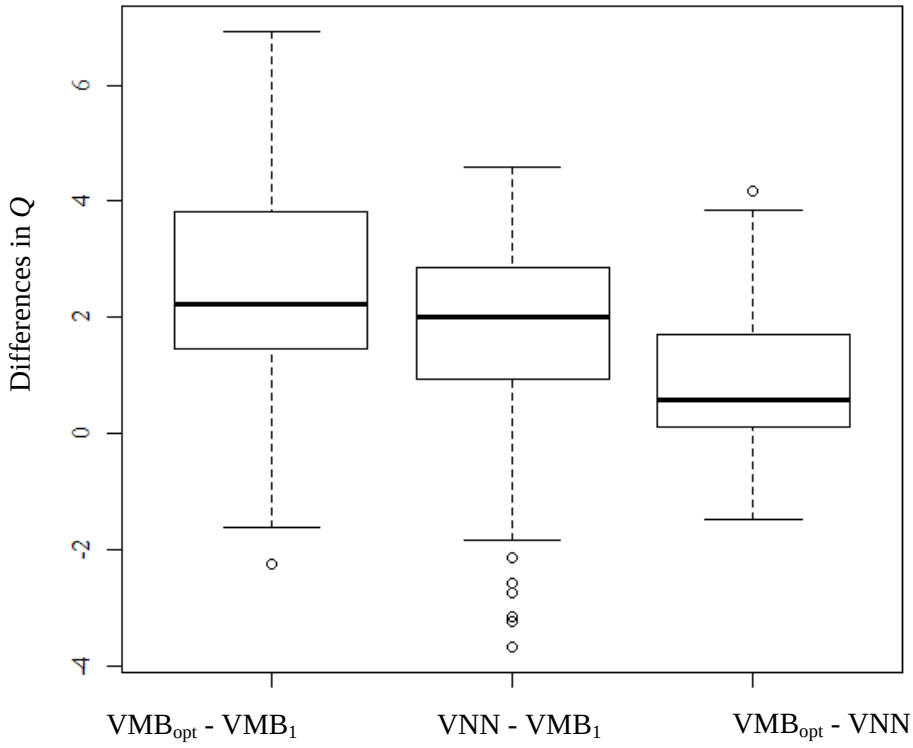


Figure 9 - Differences in Q between bootstrap methods

The overall Quade test was applied in R using the *quade.test()* function from the *stats* packages. A Quade F-value of 88.6, and a p-value of less than 2.2×10^{-16} were calculated. This p-value is much less than our cut-off of 0.05. Therefore we reject $H_{2,0}$ as expected, and provisionally conclude that there is a median difference in Q between the bootstrap methods. In other words, the bootstrap method that is used does have a statistically significant effect on performance, Q.

Next, we want to determine where these differences lie. We cannot simply apply a test for paired data (such as the Wilcoxon signed rank test) for each pair of methods, because this involves reusing data. Instead, multiple comparison tests are needed that adjust the significance level so as to reveal only *honestly significant differences* between the methods. Since R does not include a function to perform Quade multiple comparison tests, the algorithm given in Appendix C was followed. The pair-wise difference hypotheses and the test results are shown in Table 8, along with the sample median difference between each pair of methods. Because every null hypothesis was rejected, we can provisionally conclude that VMB_{opt} performs significantly better than VNN, which performs significantly better than VMB_1, when tested at the 5% level of significance. We cannot evaluate the power of the non-parametric Quade test at these sample sizes, but it is immaterial in this case, because significant differences were detected.

Table 8 - Multiple comparison tests results

Null hypothesis	Alternative hypothesis	Median difference (sample)	1-sided p-value	$\alpha/2$ (critical value)	Conclusion
$\mu_{(VMBopt\ -VMB1)} = 0$	$\mu_{(VMBopt\ -VMB1)} \neq 0$	2.2	7.3×10^{-11}	2.5×10^{-2}	Reject null hypothesis
$\mu_{(VNN-VMB1)} = 0$	$\mu_{(VMBopt\ -VMB1)} \neq 0$	2.0	3.9×10^{-4}	2.5×10^{-2}	Reject null hypothesis
$\mu_{(VMBopt\ -VNN)} = 0$	$\mu_{(VMBopt\ -VMB1)} \neq 0$	0.59	4.6×10^{-4}	2.5×10^{-2}	Reject null hypothesis

7 DISCUSSION

For the sample data from the petro-chemical plant, the number of variables in the trace series was not found to have a significant effect on the performance of any of the bootstrap methods. This result is useful in practice, because one may not be sure whether certain inputs to a simulation model are cross-correlated or not. The result tells us that if there is any doubt about the inter-dependence of the input variables, it is better to combine them into one trace series before applying a bootstrap method. Doing this will not cause a significant loss in performance regarding the preservation of auto-correlations. (Note: It will still result in a loss of variety between the variables, because they will be sampled together as a vector.)

The conclusion regarding the effect of the number of variables was marginal for the VMB_1 method. It might just have been a result of the test used and the sample size. Fortunately, this uncertainty does not matter much, because one would not use the VMB_1 method anyway. It was only tested as a control and performed worse than the other methods, as expected.

The VMB_{opt} method performed better than the VNN method. This could be because of the simplistic nearest-neighbour matching algorithm used, which only compared one previous timeframe. A different one could have been used, but the ideal matching algorithm could be different for each trace series, depending on its auto-correlation structure. The VMB_{opt} method has the advantage that it automatically adjusts itself to the auto-correlation structure of each trace series, because it uses whichever block length performs best. This 'brute force' optimization is practical for the VMB_{opt} method because it is quick enough that many block lengths can be applied and tested. It would be infeasible for the VNN method though, because it is much more computationally intensive. Another possible explanation for the better performance of the VMB_{opt} method is the general approach used. It might just be more effective to preserve auto-correlations by re-sampling data in blocks, rather than by trying to make each new re-sampled timeframe 'fit' after the previous one. Whatever the reason, the VMB_{opt} method performed best for the trace series from the petro-chemical plant, and should be used in preference.

Both the VMB_{opt} and VNN methods performed much better than the control VMB_1 method. This is not surprising. The VMB_1 method destroys the auto-correlations in the trace series by definition, because it samples individual timeframes in random order. As the block length increases, fidelity to the auto-correlation structure of the trace series increases. If the block length increases too much, though, the generated series loses variety, and starts looking too much like the trace series. The optimal block length is a compromise between the two. The effect of the block length is illustrated in the figures below. Figure 10 shows an example univariate trace series, 100 timeframes long, that has

some auto-correlation ($r_a \approx 0.7$). Figure 11 shows the generated series by the VMB_1 bootstrap method, which has a block length of one timeframe. It is clear from the pattern of the generated series that the auto-correlation of the trace series is not preserved. Figure 12 shows the generated series by the VMB bootstrap method with a block length of 25 timeframes. The auto-correlation pattern is preserved, but the generated series resembles the trace series unrealistically closely.

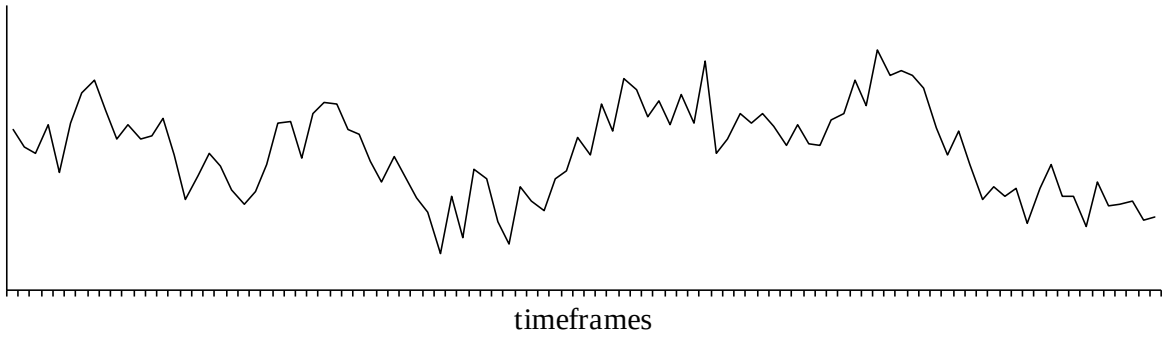


Figure 10 - Example univariate trace series

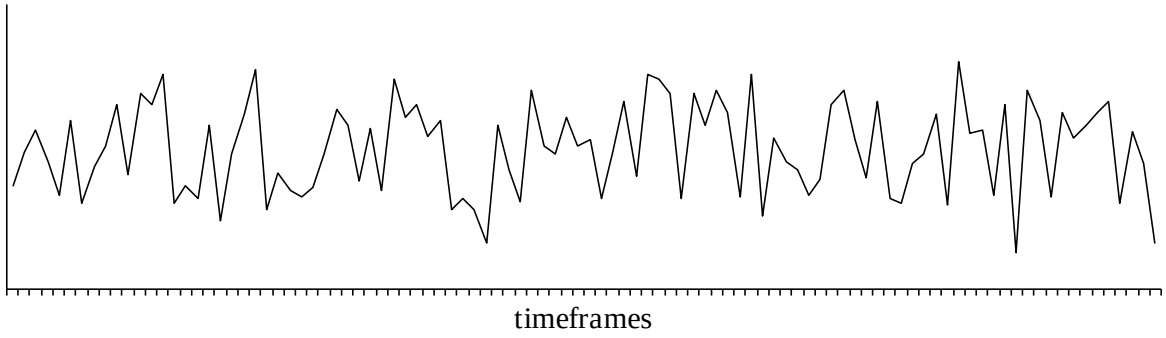


Figure 11 - Generated series by the VMB_1 bootstrap method (block length of 1 timeframe)

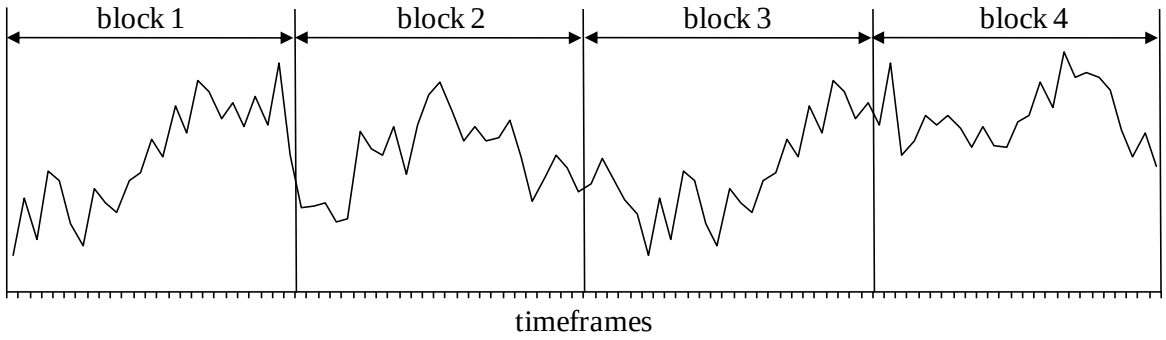


Figure 12 - Generated series by the VMB bootstrap method (block length of 25 timeframes)

The performance measure Q incorporates the compromise between fidelity and variety, and shows what the optimum block length is. Since Q is defined for each variable in a trace series individually, the overall optimal block length is that which maximizes the total Q for the trace series. Figure 13 plots Q versus block length for an example trace series that is 1000 timeframes long and contains 4 variables. Q is estimated using the subseries Turing test technique, so the block lengths shown are for a subseries (32 timeframes long), not the whole trace series. The performance of every possible block length for a subseries is shown.

The graph shows that for each variable individually, performance increases sharply as the block length increases from one timeframe to the optimum for that variable. Performance then gradually declines as block length increases beyond the optimum. Similarly, the total Q for the trace series rises sharply to its optimum, then declines gradually, as block length increases. This is typical for stationary trace series where auto-correlation is present.

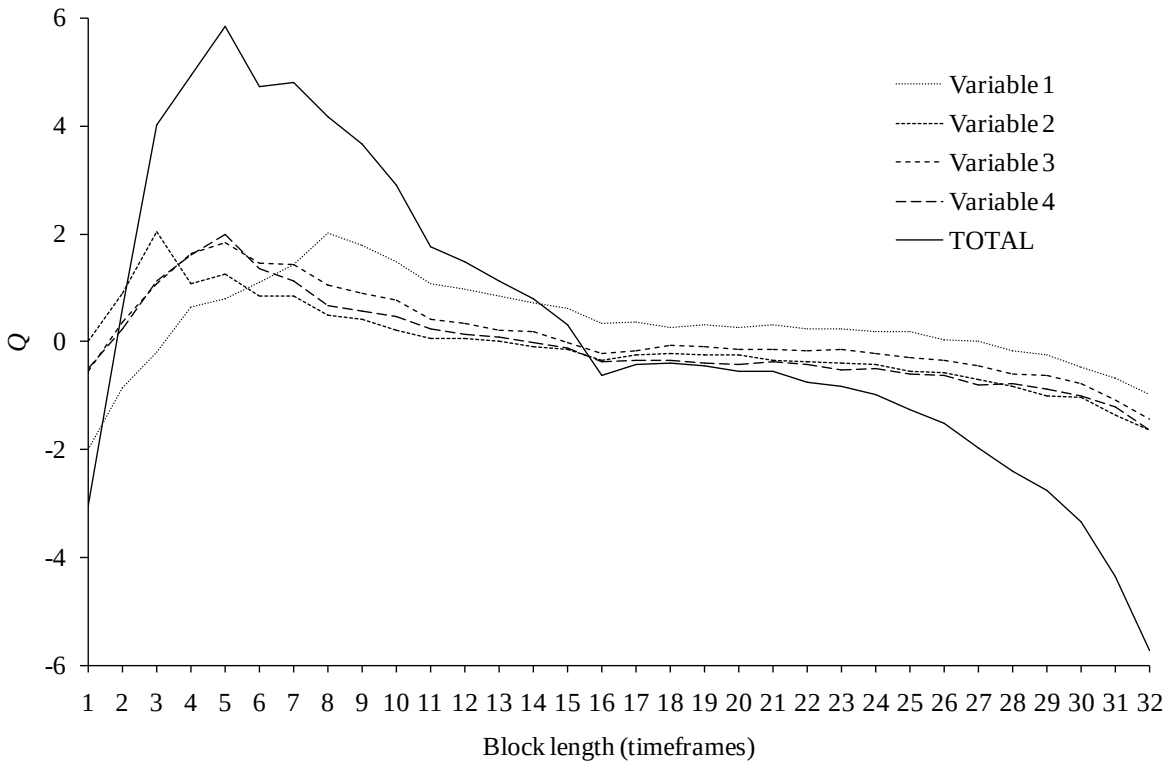


Figure 13 - Performance versus block length for an example trace subseries

Figure 13 illustrates why it is necessary to optimize the block length for each trace series. Using a block length of one timeframe is a simple but inadequate approach. The VMB_{opt} method finds the optimal block length for each trace series using the subseries Turing test procedure. This makes it more complicated and more computationally intensive than the simplistic VMB_1 bootstrap method. However, it produces generated data with much better quality. Since the quality of these input

scenarios determines the quality of simulation outputs, the VMB_{opt} method should be used in preference to the VMB_1 method at the petro-chemical plant if it is suspected that a simulation's input variables are auto-correlated.

In summary, when using the VMB_{opt} bootstrap method to generate multivariate scenarios for a simulation model, one must:

1. Assess which input variables may be auto- and/or cross-correlated, and assemble their observed data into a multivariate trace series.
2. Check that the trace series complies with the assumptions of the VMB_{opt} bootstrap method. It must contain only numerical, stationary time series with short-memory auto-correlation and simple cross-correlation.
3. Use the subseries Turing test technique to find the optimal overall block length for a subseries.
4. Scale this up for the whole trace series by multiplying it by $\sqrt{n/m}$ as prescribed by Demirel & Willemain (2002b), where n is the trace series length and m is the subseries length.
5. Apply the vector moving block bootstrap method to the trace series, using the optimal block length (VMB_{opt}), to produce the generated series, as required.

8 CONCLUSIONS

Three vector bootstrap methods were tested empirically, using data from a petro-chemical plant:

- the vector moving block bootstrap method, with a block length of 1 timeframe (VMB_1)
- the vector moving block bootstrap method, with an optimized block length (VMB_{opt})
- the vector nearest neighbour bootstrap method (VNN)

The following conclusions can be made:

- The number of variables in a trace series did not make a statistically significant difference to the performance of the bootstrap methods, at the 5% significance level.
- The VMB_{opt} method performed significantly better than the VNN method, which in turn performed significantly better than the VMB_1 method, at the 5% significance level. Because the VMB_{opt} bootstrap method produced the most realistic scenarios, it is the most suitable method for the petro-chemical plant. It should be used to generate multivariate input scenarios for simulation models if the presence of auto and cross-correlation is suspected.

9 RECOMMENDATIONS FOR FURTHER RESEARCH

Further research is needed to generalize these results beyond the sample taken from the petro-chemical plant. Artificial data could be simulated so as to cover a wide range of auto- and cross-correlation structures.

Further research is also needed to relax the assumptions of the VMB_{opt} bootstrap method, regarding:

- Data type: For trace series containing nominal variables, one would need a generalized version of the delta (Δ) measure, which quantifies the difference in the auto-correlation structures of two time series.
- Time-variation: For non-stationary trace series, one would need to work with rates of change.
- Dependencies: For trace series with long-memory auto-correlation and complex inter-dependencies between the variables, one would need more sophisticated sampling algorithms.

In addition, the 'brute force' way in which the block length of the VMB method is optimized is fairly computationally intensive. It would be useful to have a rule by which to predict the optimal block length for a trace series without performing a subseries Turing test for every possible block length. Since the optimal block length depends on the auto-correlation structures in the trace series, the rule would need to characterize these structures and relate them to the optimal block length in some way. Further research is needed to survey, develop and test potential optimization rules.

Finally, a serious limitation of vector bootstrap methods is that they cannot produce variety between the variables in the generated series. To preserve cross-correlations, they sample all the variables from the trace series as a vector. This means that the simulation model is only exposed to the limited combinations of variables that have been observed in the trace series. Further research is needed to survey, develop and test bootstrap methods that can preserve cross-correlations without sampling as a vector, so that the generated series can contain new but realistic combinations.

REFERENCES

Barton, R.R. et al., 2002. Panel discussion on current issues in input modeling. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*. San Diego, California: Winter Simulation Conference, pp. 353-369.

Biller, B., 2009. Copula-Based Multivariate Input Models for Stochastic Simulation. *Operations Research*, 57(4), pp.878-892.

Biller, B. & Nelson, B.L., 2008. Evaluation of the ARTAFIT Method for Fitting Time-Series Input Processes for Simulation. *INFORMS Journal on Computing*, 20(3), pp.485-498.

Biller, B. & Nelson, B.L., 2005. Fitting time-series input processes for simulation. *Operations Research*, 53(3), pp.549–559.

Borkowski, J., 2010. Course notes: Nonparametric Statistics - Quade test, *Department of Mathematical Sciences, Montana State University*. Available at: http://www.math.montana.edu/~jobo/st431/quade.pdf [Accessed October 18, 2011].

Branca, F., 2011. Discussion concerning sampling trace series from the petro-chemical plant.

Cario, M.C. & Nelson, B.L., 1997. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. *Northwestern University, IEMS Technical Report*.

Carlstein, E., 1986. The use of subseries methods for estimating the variance of a general statistic from a stationary time series. *Annals of Statistics*, 14, pp.1171-1179.

Carlstein, E. et al., 1998. Matched-Block Bootstrap for Dependent Data. *Bernoulli*, 4(3), pp.305-328.

Civelek, I., Biller, B. & Scheller-Wolf, A., 2009. The Impact of Dependence on Queueing Systems. *Tepper School of Business Carnegie Mellon University, Pittsburgh, USA*.

Conover, W.J., 1999. *Practical Nonparametric Statistics* 3rd ed., Wiley.

Demirel, O.F. & Willemain, T.R., 2002a. A Turing Test of Bootstrap Scenarios. *Journal of Computational and Graphical Statistics*, 11(4), pp.896-909.

Demirel, O.F. & Willemain, T.R., 2002b. Generation of simulation input scenarios using bootstrap methods. *Journal of the Operational Research Society*, 53(1), pp.69-78.

Derrac, J. et al., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1, pp.3-18.

Efron, B., 1979. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pp.1–26.

Hall, P., Horowitz, J.L. & Jing, B.Y., 1995. On blocking rules for the bootstrap with dependent data. *Biometrika*, 82(3), p.561.

Kunsch, H.R., 1989. The Jackknife and the Bootstrap for General Stationary Observations. *The Annals of Statistics*, 17(3), pp.1217-1241.

Lahiri, S.N., 2004. Selecting optimal block lengths for block bootstrap methods. *Department of Statistics, Iowa State University*. Available at:

http://neptune.galaxy.gmu.edu/interface/I03/I2003Proceedings/LahiriSoumendra/LahiriSoumendra.paper.pdf [Accessed September 14, 2011].

Lahiri, S.N., 1999. Theoretical Comparisons of Block Bootstrap Methods. *The Annals of Statistics*, 27(1), pp.386-404.

Lall, U. & Sharma, A., 1996. A nearest neighbor bootstrap for resampling hydrologic time series. *Water Resources Research*, 32(3), pp.679–693.

Lee, T. & Ouarda, T.B.M.., 2011. Identification of model order and number of neighbors for k-nearest neighbor resampling. *Journal of Hydrology*.

Liu, R.Y. & Singh, K., 1992. Moving blocks jackknife and bootstrap capture weak dependence. *Exploring the limits of bootstrap*, 225, p.248.

Park, D. & Willemain, T.R., 1999. The threshold bootstrap and threshold jackknife. *Computational Statistics & Data Analysis*, 31(2), pp.187–202.

Politis, D.N. & Romano, J.P., 1992. A circular block-resampling procedure for stationary data. *Exploring the limits of bootstrap*, pp.263–270.

Politis, D.N. & Romano, J.P., 1994. The Stationary Bootstrap. *Journal of the American Statistical Association*, 89(428).

Rajagopalan, B. & Lall, U., 1999. A k-nearest-neighbor simulator for daily precipitation and other weather variables. *Water Resources Research*, 35(10), pp.3089–3101.

Teknomo, K., 2006. Similarity Measurement. Available at: http://people.revoledu.com/kardi/tutorial/Similarity/ [Accessed March 2, 2011].

Willemain, T.R., Bress, R.A. & Halleck, L.S., 2003. Enhanced Simulation Inference Using Bootstraps of Historical Inputs. *IIE - Transactions*, 35(9), p.851.

Zientek, L.R. & Thompson, B., 2007. Applying the bootstrap to the multivariate case: bootstrap component/factor analysis. *Behavior Research Methods*, 39(2), pp.318-325.

APPENDIX A: DIGITAL APPENDIX

The compact disc contains:

- *trace_series.xls* – the 28 trace series observed from the petro-chemical plant in a spreadsheet
- *trace1.txt* to *trace28.txt* – the 28 observed trace series in a text files, ready to be used by the R code
- *code.txt* – the R code used to run and test the bootstrap methods

APPENDIX B: R CODE FOR BOOTSTRAP METHODS

```
######################### DELTA quality measure function ##########################
DELTA_calc=function(series1,series2){
	#series1
	#series2
	series_length=dim(series1)[1]
	no_vars=dim(series1)[2]

	HOC1=matrix(NA,10,no_vars)
	HOC2=matrix(NA,10,no_vars)
	# for each variable in the series
	for(var_no in 1:no_vars){
		uni1=series1[,var_no]-mean(series1[,var_no])
		uni2=series2[,var_no]-mean(series2[,var_no])
		#for each higher order difference series
		for (diff_no in 1:10){
			#HOC for series 1
			HOC1[diff_no,var_no]=sum((uni1[2:length(uni1)]*
				uni1[1:(length(uni1)-1)])<0)
			if (HOC1[diff_no,var_no]==0) {HOC1[diff_no,var_no]=1}
			if (HOC1[diff_no,var_no]==(length(uni1)-1)) {HOC1[diff_no,
				var_no]=HOC1[diff_no,var_no]-1}
			HOC1[diff_no,var_no]=HOC1[diff_no,var_no]/(length(uni1)-1)
			uni1=uni1[2:length(uni1)]-uni1[1:(length(uni1)-1)]
			#HOC for series 2
			HOC2[diff_no,var_no]=sum((uni2[2:length(uni2)]*uni2[1:
				(length(uni2)-1)])<0)
			if (HOC2[diff_no,var_no]==0) {HOC2[diff_no,var_no]=1}
			if (HOC2[diff_no,var_no]==(length(uni2)-1)) {HOC2[diff_no,
				var_no]=HOC2[diff_no,var_no]-1}
			HOC2[diff_no,var_no]=HOC2[diff_no,var_no]/(length(uni2)-1)
			uni2=uni2[2:length(uni2)]-uni2[1:(length(uni2)-1)]
		}
	}
	#calculate delta
	DELTA=colSums(((HOC1-HOC2)/sqrt((HOC1*(1-HOC1)+HOC2*(1-HOC2))/
		(series_length-1:10)))^2)
	DELTA
}

######################### Vector Nearest Neighbour Bootstrap ########################
VNN = function (trace,req_gen_length,K=sqrt(dim(trace)[1])){
	#trace
	#req_gen_length
	#L=maximum lag for comparison
	#K=number of candidates
	trace_length=dim(trace)[1]
	no_vars=dim(trace)[2]


	#determine lag 1 auto-correlation for each variable
	auto_corr=rep(NA,no_vars)
	for (var_no in 1:no_vars) {auto_corr[var_no]=abs(cor(method="spearman",
		trace[,var_no],c(trace[trace_length,var_no],trace[1:(trace_length-1),
		var_no])))}

	#define gen and sample a timeframe to get things going (warm-up period)
	gen=matrix(NA,req_gen_length,no_vars)
	gen[1,]=trace[sample(1:trace_length,1),]

	#for each NN bootstrap sample
	for (samp_no in 2:req_gen_length){
		#compare feature vector with every timeframe in the trace
```

```
            diff=abs(trace-rep(gen[samp_no-1,],each=trace_length))
            #for each variable, rank across all timeframes
            for (var_no in 1:no_vars) {diff[,var_no]=rank(diff[,var_no],
                  ties.method="random")}
            #for each timeframe, weight the ranks (weighting by autocorrelation)
            for (time_no in 1:trace_length) {diff[time_no,]=diff[time_no,]*
                  auto_corr}
            #for each timeframe, sum across the weighted ranks
            diff=rowSums(diff)
            #rank across all timeframes
            diff=rank(diff,ties.method="random")
            #sample from the best K timeframes
            samp=sample(1:K,1,prob=1/(1:K)/sum(1/(1:K)))
            samp=which(diff==samp)
            #add to the generated data
            gen[samp_no,]=trace[samp,]
      }
      gen
}

############################## Moving Block Bootstrap ##############################
VMB=function(trace,req_gen_length,block_length=0){
      #trace
      #req_gen_length
      #block_length = specified length of sampling block
      trace_length=dim(trace)[1]
      no_vars=dim(trace)[2]
      #generate bootstrap series
      start_rows=sample(x=1:trace_length,size=req_gen_length%/%block_length+1,
            replace=TRUE)
      sample_rows=rep(start_rows,each=block_length)+0:(block_length-1)
      sample_rows[sample_rows>trace_length]=sample_rows[sample_rows>trace_length]-
            trace_length
      gen=(trace[sample_rows,])[1:req_gen_length,]
      gen
}

################################### Turing Test ###################################

D=list("Output - D")
Delta_true=list("delta_true")
Delta_VNN=list("delta_VNN")
Delta_VMB=list("delta_VMB")

for(test_no in 1:(2)){
      trace=as.matrix(read.table(paste("trace",test_no,".txt",sep=""),
            header=TRUE))[1:1000,]
      #standardise trace
      trace=scale(trace)
      trace_length=dim(trace)[1]
      no_vars=dim(trace)[2]
      no_subs=min(100,floor(sqrt(trace_length)))
      subs_length= min(100,trace_length%/%no_subs)

      #estimate the true distribution of DELTA for each variable
      #enumerate all the possible combinations of subseries
      combinations=cbind(rep(1:no_subs,each=no_subs),rep(1:no_subs,times=no_subs))
      combinations=combinations[combinations[,1]<combinations[,2],]
      #for each combination calculate and store DELTA for each variable
      no_comb=no_subs*(no_subs-1)/2
      DELTA_TRUE=matrix(NA,no_comb,no_vars)
      for (comb_no in 1:no_comb){
            trace_subs_no=combinations[comb_no,1]
            add_subs_no=combinations[comb_no,2]
            trace_subs=trace[((trace_subs_no-1)*subs_length+1):(trace_subs_no*
```

```
                subs_length),]
        add_subs=trace  [((add_subs_no-1)  *subs_length+1):(add_subs_no*
                subs_length),]
        DELTA_TRUE[comb_no,]=DELTA_calc(trace_subs,add_subs)
    }


    #estimate the distribution of DELTA generated by the
        #Vector Nearest Neighbour method for each variable
    DELTA_VNN=matrix(NA,no_subs*no_subs,no_vars)
    j=0
    for (subs_no in 1:no_subs){
        print(paste("test",test_no, "of 28","method 1 of 2,",
                floor(subs_no/no_subs*100),"% done"))
        flush.console
        trace_subs=trace[((subs_no-1)*subs_length+1):(subs_no*subs_length),]
        #generate many new series, and compare
        for (i in 1:no_subs){
            j=j+1
            #generate a new series
            gen_subs=VNN(trace=trace_subs,req_gen_length=subs_length)
            #compare the trace subseries to the generated subseries
            DELTA_VNN[j,]=DELTA_calc(trace_subs,gen_subs)
        }
    }

    #estimate the distribution of DELTA generated by the vector moving block
        #bootstrap method
    DELTA_VMB=rep(NA,no_subs*no_subs*no_vars*subs_length)
    dim(DELTA_VMB)=c(no_subs*no_subs,no_vars,subs_length)
    for (block_length in 1:subs_length){
        print(paste("test",test_no, "of 28","method 2 of 2,",
                floor(block_length/subs_length*100),"% done"))
        flush.console
        j=0
        for (subs_no in 1:no_subs){
            trace_subs=trace[((subs_no-1)*subs_length+1):
                    (subs_no*subs_length),]
            #generate many new series, and compare
            for (i in 1:no_subs){
                j=j+1
                #generate a new series
                gen_subs=VMB(trace=trace_subs,req_gen_length=
                        subs_length,block_length=block_length)
                #compare the trace subseries to the generated subseries
                DELTA_VMB[j,,block_length]=
                        DELTA_calc(trace_subs,gen_subs)
            }
        }
    }

    #do ks tests and compile results
    D_VNN=rep(NA,no_vars)
    D_VMB=matrix(NA,subs_length,no_vars)
    for(var_no in 1:no_vars){
        D_VNN[var_no]=ks.test(DELTA_TRUE[,var_no],DELTA_VNN[,var_no])[[1]]
        for (block_length in 1:subs_length){
            D_VMB[block_length,var_no]=ks.test(DELTA_TRUE[,var_no],
            DELTA_VMB[,var_no,block_length])[[1]]
        }
    }
    D=c(D,list(rbind(D_VNN,D_VMB)))
}

#convert these D values to Q values with Q=log((1-D)/D)
```

APPENDIX C: QUADE MULTIPLE COMPARISON TESTS

Non-parametric Quade multiple comparison tests were performed according to the following algorithm, adapted from Derrac *et al* (2011, pp.8,10) and Borkowski (2010). We have g blocks (the experimental subjects) and c treatments (the bootstrap methods), with every treatment applied once to every block, as shown in Table 9.

Table 9 - Within-subjects data format

Block (subject)	Treatment (within-subjects factor)			
	1	2	...	c
1	x_{11}	x_{12}	...	x_{1c}
2	x_{21}	x_{22}	...	x_{2c}
3	x_{31}	x_{32}	...	x_{3c}
...	...	...	...	...
g	x_{g1}	x_{g2}	...	x_{gc}

Test whether the difference between two treatments (m and n) is statistically significant, as follows:

- Rank the observations within each block (each row), from 1 to c. For tied values, assign average ranks. Let R_{ij} be the rank assigned to block i and treatment j.
- Calculate the sample range of each block ($\text{row max}\, x_{ij} - \text{row min}\, x_{ij}$). Rank the rows according to this sample range, in ascending order, assigning average ranks to ties. Let P_i be the rank assigned to block i and treatment j.
- Multiply each ranked observation by the ranked range of its block, $S_{ij} = P_i[R_{ij}]$.
- Sum this product for each treatment (column), $S_j = \sum_{i=1}^{g} S_{ij}$ for $j = 1, 2, \ldots, c$.
- Calculate: $A = \sum_{i=1}^{g} \sum_{j=1}^{c} S_{ij}^2$
- Calculate: $B = 1/g \left[\sum_{j=1}^{c} S_j^2\right]$
- For a pair of treatments m and n, calculate: $t_{mn} = \frac{|S_m - S_n|}{\sqrt{\frac{2g(A-B)}{(g-1)(c-1)}}}$
- Perform a one-tailed *t*-test on this value at the required level of significance (α). If the p-value is less than ($\alpha/2$), reject the null hypothesis that there is no pair-wise difference between the treatments.